REMARKS/ARGUMENTS

This letter is in reply to the Final Office Action dated May 13, 2009.

Claim Amendments

To expedite prosecution of the application, claims 1, 5, 7, 11, 13, 14, 16, 17 and 19-21 have been amended. Claims 3, 9, 15 have been cancelled, without prejudice. New claims 22-24 have been added. Accordingly, **claims 1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17 and 19-24** remain in this application. Claims 1, 7 and 12 are independent claims.

Applicants have amended claim 1 to include certain features previously recited in claim 3, for example. Claim 1 has also been amended to recite that two or more files are generated from a plurality of class files. Support for this amendment can be found at paragraph 31, for example, in the Applicants' description as filed.

Applicants have made amendments to claims 7 and 12 that are analogous to the amendments made to claim 1.

Applicants have also made minor amendments to claims 5, 11, 17 and 19-21 for consistency with the amendments made to the independent claims.

Applicants have also amended claims 13-17 and 21 to use the term "storage medium" rather than the term "computer readable medium" as suggested by the Examiner.

New claim 22 recites that as few sibling files are generated as possible while satisfying a constraint on individual generated file size. Support for this claim may be found at, for example, lines 4 to 5 in paragraph 32 of the Applicant's description as filed. Claims 23 and 24 recite analogous features.

Objections to the Disclosure

In section 9 of the Office Action, the Examiner objected to the disclosure because claims 13-17 and 21 are drawn towards a "computer-readable medium" and the Examiner believes that there is no support for this term in the specification. The Examiner suggested that there is support for the term "storage medium" and requested appropriate correction.

Without prejudice, Applicants have amended claims 13, 14, 16, 17 and 21 to use the term "storage medium" instead of the term "computer-readable medium" as per the Examiner's suggestion. Withdrawal of this Examiner's objection is respectfully requested.

Claim Rejections - 35 U.S.C. §103

Claims 1-5, 7-11, 13-17 and 19-21 stand rejected under 35 U.S.C. §103(a) as being unpatentable over WO 99/49392 in the name of Baentsch et al. ("Baentsch") in view of U.S. Publication No. 2002/0170047 A1 in the name of Swetland ("Swetland"). The Applicants respectfully traverse all rejections.

As will be noted below, Applicants respectfully submit that the person skilled in the art would not be motivated to combine the teachings of the relied-upon documents to arrive at the claimed subject matter.

However, to expedite prosecution of the application, and without prejudice, Applicants have amended claim 1 to recite additional features that are clearly neither taught nor suggested by the relied upon documents, for example, that at least two of the generated files are generated as sibling files in a common sibling group, wherein each of the sibling files comprises a sibling list for listing other sibling files in the common sibling group, wherein cross-references between the sibling files in the common sibling group

are indicated using hard offsets, and wherein references to files that are not part of the common sibling group are indicated using symbolic references.

Applicants submit that the use of sibling files allows multiple generated files to be related to one another so that hard offsets may be used for cross-referencing certain elements between related sibling cod files. This provides a number of advantages. Firstly, as stated in paragraph 24 on page 5 of the application as filed, using hard offsets directly in a generated file provides a more compact representation. In addition, as stated in paragraph 29 on pages 6 and 7 of the application as filed, sometimes there are significant limitations on the size of a single generated file. It can be advantageous to partition multiple class files into two or more sibling files, so that the size of each sibling file is smaller than the file size limit and the cumulative size of the sibling files is smaller than the cumulative size of the original class files. For example, packing class files into two sibling files will be a more compact representation than packing the class files into three generated files, since some of the symbolic references for cross-references between classes in the three generated files can be replaced with hard offsets in the two sibling files which requires less memory (see e.g. paragraphs 30-32 on pages 7 and 8 of the application as filed).

On pages 7 and 8 of the Office Action, the Examiner argued that sibling files are obvious based on the combination of Baentsch and Swetland. Applicants respectfully disagree. Each of the Examiner's comments will now be addressed.

The teachings of Baentsch

The Examiner first argued that Baentsch discloses:

> - *wherein cross-references between the sibling files in the common sibling group are indicated using hard offsets* (see at least Page 4, lines 12-14, "The fixup table ... contains the position in the text or data section where a relocation has to take place. In the simple case, these places are also relocated by a precalculated offset into well known trusted packages.")

> - *and references to files that are not part of the common sibling group are indicated using symbolic references* (see at least Page 4, lines 22-24, "... references to other external packages should not be linked by precalculated offsets. Instead, a name or identifier should be used for references to other packages during the link process." These references made to external packages can include methods.)

In response, Applicants submit that Baentsch teaches the technique of using precalculated offsets or symbolic references for resolving references during *linking*, not for **generating** files such that the file size of the generated files is reduced compared to the original files. For instance, Baentsch clearly states that "the resolving of references (linking) is the subject of the present invention" (see lines 21 on page 1 of Baentsch) and that "an object of the present invention is to achieve efficient linking in resource constraint Java runtime environments" emphasis added by Applicants (see lines 23-24 on page 1 of Baentsch). Applicants note that Baentsch refers to linking and runtime, not to storage of files. Furthermore, Baentsch teaches that the references are resolved using precalculated offsets for well-known and trusted packages, and using names or identifiers for external packages. These packages are clearly software library files that are generated by other parties; the well-known and trusted packages are IBM or JavaSoft software library files (see lines 24-31 on page 3 of Baentsch) while the external packages are most likely software libraries developed by third parties. The references are resolved *after* the file that is being resolved is generated; the references are not being defined while the file is being **generated** as is taught and claimed in Applicants' claim 1.

Furthermore, Baentsch explicitly describes how these references are used during linking (see lines 6 to 22 on page 3 of Baentsch), and that linking is not flexible enough to allow different implementations of system classes and specified extensions on different cardlets (see lines 24 to 31 on page 3 of Baentsch). This problem does not occur for the Applicants' device since the hard offsets and symbolic references are used

directly in the generated files as recited in Applicants' claim 1, in contrast to Baentsch which teaches resolving these references only during linking.

Therefore, Applicants submit that Baentsch does not teach using hard offsets in generated files for reducing the file size of generated files as recited in Applicants' claim 1. Applicants further submit that the elements relied upon by the Examiner in Baentsch are used for a different purpose (i.e. linking) than that recited in Applicants' claim 1 (i.e. reducing storage requirements) and cannot provide the same benefits as the generated files recited in Applicants' claim 1.

The teachings of Swetland

The Examiner then argued that Swetland discloses:

> - *wherein there are at least two generated files defined as sibling files in a common sibling group, each of the sibling files comprising a sibling list for listing other sibling files in the common sibling group* (see at least Figure 12; Paragraph 0083, "... each of the class files used in a particular application program ... are combined to form a unified programming object referred to herein as a 'bundle'. For the purpose of illustration, the particular bundle ... is constructed from the class files." Related class files are combined to form a bundle, which one of ordinary skill in the art would view as a sibling group.)

In response, Applicants submit that Figure 12 in Swetland shows multiple class files that are mapped to a bundle that is clearly a single file, since the bundle 1200 in Figure 12 has the exact overall structure as the class files 200 and 230. Also, it is well known that the term "bundle" in software programming means to combine elements into one package: in this case, a single file.

Furthermore, Swetland teaches that the class files are combined to form a unified programming object (see lines 1 to 4 in paragraph 83 in Swetland). Those of general skill in the art of object oriented programming realize that an object is typically defined

by a class and the class is stored in a single file. Also, Swetland does not combine the class files by simply packing the files together into the bundle. Swetland teaches stripping out the duplicated components of the class files when generating the bundle file to create a shared constant pool to remove redundant pool entries that would occur from storing the class files independently (see paragraphs 10 to 12 in Swetland), which also implies that the bundle is a single file.

Even if one considered Swetland's bundle to be the same as Applicants' sibling group, according to the Examiner's interpretation, Swetland is only combining the original class files into the bundle. This is different than the Applicants' claim 1, which recites that two or more files are generated from the original class files such that there are at least two sibling files.

In addition, Applicants' amended claim 1 recites that each sibling file includes a sibling list that lists other sibling files in the same sibling group. Swetland does not teach providing a sibling list in the bundle. Rather, Swetland simply teaches that the bundle has a header 1201, a shared constant pool 1202, and a section for the methods and fields from the original class files. Accordingly, there is no need to have a sibling list that lists other related files in Swetland's bundle since Swetland teaches combining all of the class files into a single file, i.e. the "bundle". Furthermore, while Swetland does teach copying the header data from the original class files into the bundle, this header data is for the original class files themselves, not for newly generated sibling files which are generated from the original class files as recited in Applicants' amended claim 1.

Accordingly, one cannot make any assumptions that the bundle contains multiple files because Swetland does not clearly show this. On the contrary, the generated structure taught by Swetland more closely resembles a single class file. Therefore, it is clear that Swetland does not teach generating at least two sibling files from a plurality of class files as recited in Applicants' claim 1, but rather suggests packaging multiple class files into one file.

The skilled person would not be motivated to combine the teachings of Baentsch and Swetland

The Examiner further argued:

> "Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to incorporate Swetland's bundled class files into Baentsch's fixup table containing references to the related files. The modification would be obvious because one of ordinary skill in the art would be motivated to group related class files to form a unified class structure in a medium that limits the size of an individual file by having multiple smaller related files as opposed to a single larger combined file."

In response, Applicants respectfully submit that, while Swetland may want to provide a system and method for reducing the memory requirements for object oriented programming, Swetland teaches that this goal is achieved in a different way: by compressing the class files into one unified programming object by eliminating redundant entries in a shared constant pool (see lines 7-10 in paragraph 83 in Swetland) and copying the methods, fields and header data from the original class files (see paragraph 84 in Swetland). Swetland does not disclose that there is a need for other ways to reduce file size, and Swetland does not even recognize the problem of a limit on single file size. There are also no other passages in Swetland that would motivate a skilled person in the art to look for other ways of generating a file from a plurality of class files in order to reduce storage requirements for the generated files.

Furthermore, Applicants submit that there is no need to combine Baentsch's fixup table with Swetland bundle since Swetland teaches that referencing occurs within the bundle itself with respect to the top of the constant pool or by using other offsets in the constant pool entries to indicate where the code referenced by an entry resides within the bundle (see lines 4 to 15 in paragraph 85 of Swetland). Accordingly, Swetland teaches providing references to locations within the bundle, not to locations outside of the bundle since the bundle format described and used in Swetland is an extension of the

existing class file format and therefore contains no explicit fixup information. In contrast, Baentsch teaches providing references to packages external to a file using precalculated offsets for well-known and trusted packages and using names or identifiers for external packages. These two referencing techniques are mutually exclusive of one another. Accordingly, the skilled person, not having the benefit of the Applicants' teachings, would not recognize a need to use a fixup table with the technique taught by Swetland.

Therefore, Applicants submit that it appears that the relied upon documents fail to provide a rationale to combine the Swetland and Baentsch references since they each teach techniques that are self-contained and mutually exclusive of one another. Furthermore, even if the references were combined, Applicants submit that they still do not teach the features of the sibling files and sibling groups recited in Applicants' claim 1.

Conclusion

In view of the foregoing, the Applicants respectfully submit that amended claim 1 recites subject matter that is both novel and inventive over the cited references, and is now in form for allowance. Furthermore, since claims 7 and 13 recite analogous features, and the remaining claims are dependent, either directly or indirectly, on one of independent claims 1, 7 and 13, it is respectfully submitted that the subject matter of these claims is also novel and non-obvious for at least the same reasons. Therefore, withdrawal of the rejections under 35 U.S.C. §103 is respectfully requested.

As previously noted, it is respectfully submitted that the cited documents do not teach all the features of the amended independent claims. Furthermore, they fail to provide a teaching, suggestion or motivation to combine features of the documents to arrive at the subject matter of the amended independent claims. The Applicants note that the

Supreme Court's KSR decision[1] did not reject the use of a "teaching, suggestion or motivation" analysis as part of an obviousness analysis. The Supreme Court characterized the analysis as a helpful insight. It is respectfully submitted that the absence of a teaching, suggestion or motivation is a significant point in the Applicants' favor, as this absence is indicative of non-obviousness.

Although the Supreme Court did not reject use of a "teaching, suggestion or motivation" analysis, the Supreme Court did say that it was not the only possible analysis of an obviousness question. In the event that the Examiner chooses to pursue a different avenue for rejection, the Examiner is invited to explicitly identify the rationale and articulate the reasons on which such rejection is based, and it should be noted that any new avenue would be a new ground for rejection not due to any action by the Applicants.

The Applicants further respectfully remind the Examiner that, even after KSR, the following legal principles are still valid, having been endorsed by the Supreme Court or having been unaffected by its decision: (1) the USPTO still has the burden of proof on the issue of obviousness; (2) the USPTO must base its decision upon evidence, and it must support its decision with articulated reasoning; (3) merely demonstrating that all elements of the claimed invention exist in the prior art is not sufficient to support a determination of obviousness; (4) hindsight has no place in an obviousness analysis; and (5) Applicants are entitled to a careful, thorough, professional examination of the claims.

[1] *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398 (2007).

In view of the foregoing comments, it is respectfully submitted that the pending claims in the subject application are now in condition for allowance, and a notice to that effect is respectfully requested.

Respectfully submitted,

BERESKIN & PARR LLP/S.E.N.C.R.L., s.r.l.
Agent for the Applicants

By _____
Kendrick Lo
Reg. No. 54,948
Tel: 416-364-7311